**Lernnachweis zu Kompetenz C1G: Refactoring-Techniken für lesbaren Code**

**Kompetenz:** Ich kann einige Refactoring-Techniken aufzählen, die einen Code lesbarer und verständlicher machen.

In dieser Lernsequenz habe ich verschiedene Refactoring-Techniken kennengelernt, die die Lesbarkeit und Verständlichkeit des Codes verbessern. Dazu gehören:

1. **Extrahieren von Funktionen:** Aufteilen von komplexem Code in kleinere, benannte Funktionen für bessere Wartbarkeit.
2. **Variablenbenennung verbessern:** Klare und aussagekräftige Namen für Variablen wählen, um den Code verständlicher zu machen.
3. **Entfernen von Redundanz:** Eliminieren von überflüssigem Code, um die Klarheit zu fördern.
4. **Kommentare hinzufügen:** Erläuternde Kommentare einfügen, um komplexe Teile des Codes zu erklären.

Diese Techniken tragen dazu bei, dass der Code leichter zu lesen, zu verstehen und zu pflegen ist.

**Umsetzung im Code mit Flask:**

Hier habe ich nichts im Code umgesetzt, da dies das Lernziel der nächsten Kompetenz ist.

**Reflexion:** Die Refactoring-Techniken sind entscheidend für die Codequalität. Durch kluges Refactoring wird der Code wartbarer und leichter zu verstehen, was die Zusammenarbeit im Team und die langfristige Pflege des Projekts unterstützt.

**Lernnachweis zu Kompetenz C1F: Verbesserung der Lesbarkeit durch Refactoring**

**Kompetenz:** Ich kann mit Refactoring-Techniken einen Code lesbarer und verständlicher machen.

In dieser Lernsequenz habe ich Refactoring-Techniken angewendet, um die Lesbarkeit eines Beispielcodes zu verbessern:

**Beispiel vor dem Refactoring: Beispiel nach dem Refactoring:**

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

**Umsetzung im Code mit Flask:**

**Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung**

**Reflexion:** Refactoring-Techniken haben dazu beigetragen, den Code klarer und lesbarer zu gestalten. Die Extraktion von Funktionen, eine klare Benennung von Variablen und die Entfernung von Redundanzen tragen dazu bei, dass der Code leichter zu verstehen und zu pflegen ist. Dies fördert eine bessere Zusammenarbeit im Team und eine nachhaltige Codebasis.

**Lernnachweis zu Kompetenz C1E: Evaluation von Refactoring-Auswirkungen**

**Kompetenz:** Ich kann die Auswirkungen des Refactorings auf das Verhalten des Codes einschätzen und sicherstellen, dass das Refactoring keine unerwünschten Nebeneffekte hat.

**Evaluierung der Auswirkungen:**

* Vor dem Refactoring habe ich den bestehenden Code sorgfältig analysiert und die Bereiche identifiziert, die verbessert werden können.
* Während des Refactorings habe ich mich auf klare Benennung, Entfernung von Redundanzen und verbesserte Struktur konzentriert.
* Nach dem Refactoring habe ich den Code getestet, um sicherzustellen, dass alle Funktionen wie erwartet arbeiten und dass keine neuen Fehler eingeführt wurden.

**Reflexion:** Die Evaluierung der Refactoring-Auswirkungen erfordert eine sorgfältige Überwachung des Codes, um sicherzustellen, dass die beabsichtigten Verbesserungen erreicht wurden. Dabei ist wichtig, dass das Refactoring nicht zu ungewollten Veränderungen führt. Diese Praxis fördert nicht nur sauberen Code, sondern auch die Sicherheit und Stabilität der Anwendung.